



# An Efficient Fault-Tolerant VLSI Architecture Using Parallel Evolvable Hardware Technology

Authors: Evangelos Stefanatos & Tughrul Arslan

**System Level Integration Group**

**School of Engineering & Electronics**

**University of Edinburgh**



# Fault Tolerance using Evolvable Hardware

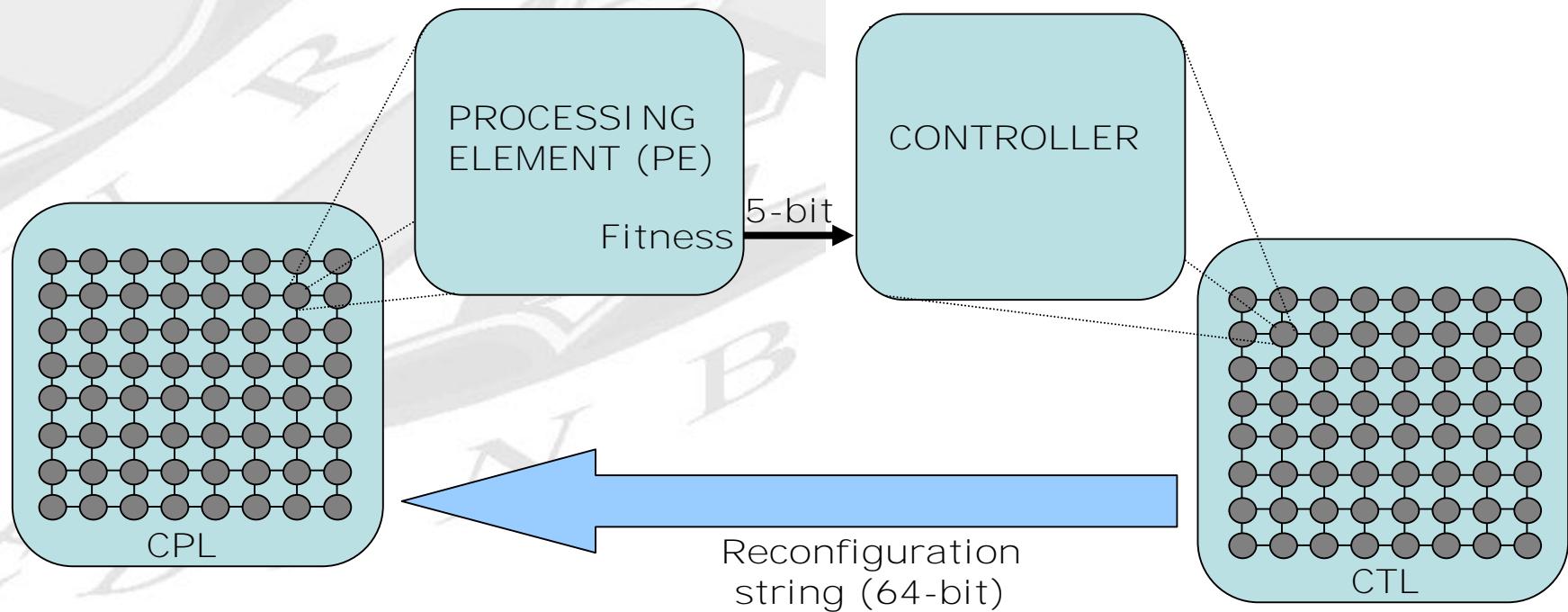
- § Automatic reconfiguration of systems operating in harsh environments, where human intervention is impossible
- § High-flexibility through reconfiguration at different levels of abstraction
- § Reduced fabrication cost due to lower redundancy
- § Achieve real-time optimization of complex problems
- § Better computational performance due to the absence of offline built-in self test (BIST) methodologies

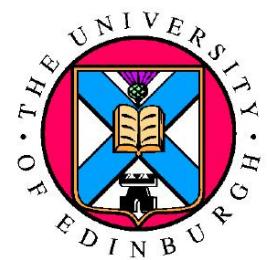
# Objectives & Characteristics



- § This work targets an entirely fault-tolerant system able to maintain its functionality in the presence of hardware faults
- § The Computational Layer (CPL) employs differential G.P.S measurements to determine the attitude of a vehicle
- § The Control Layer (CTL) comprises the fault-recovery mechanism of the system

# System Description



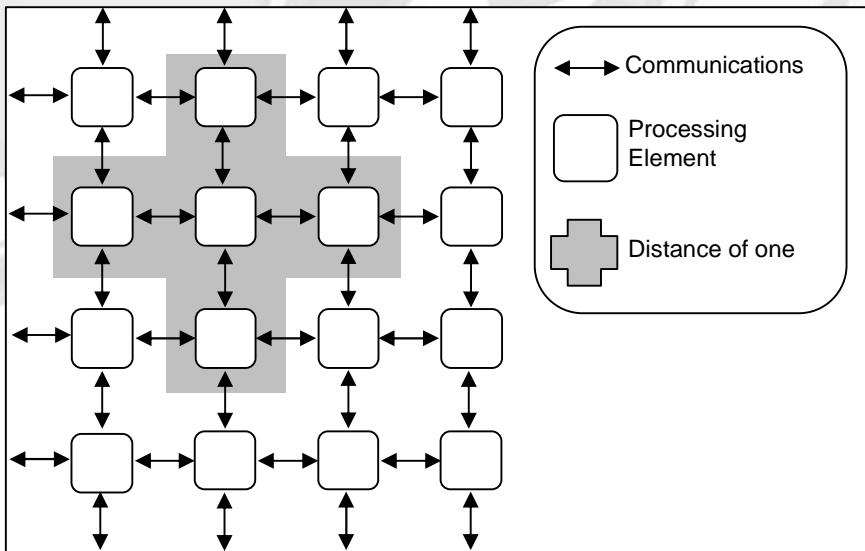


# Description of the Computational Layer (CPL)

- § GPS receivers present superiority over gyros in terms of cost and reliability
- § CPL uses the Ambiguity-Function-Method (AFM) to solve the ambiguity problem
- § The 64 PEs comprise a Parallel, Fine-grained Genetic Algorithm (PFGA)

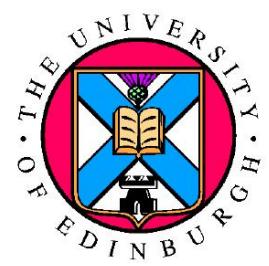


# PFGA Architecture



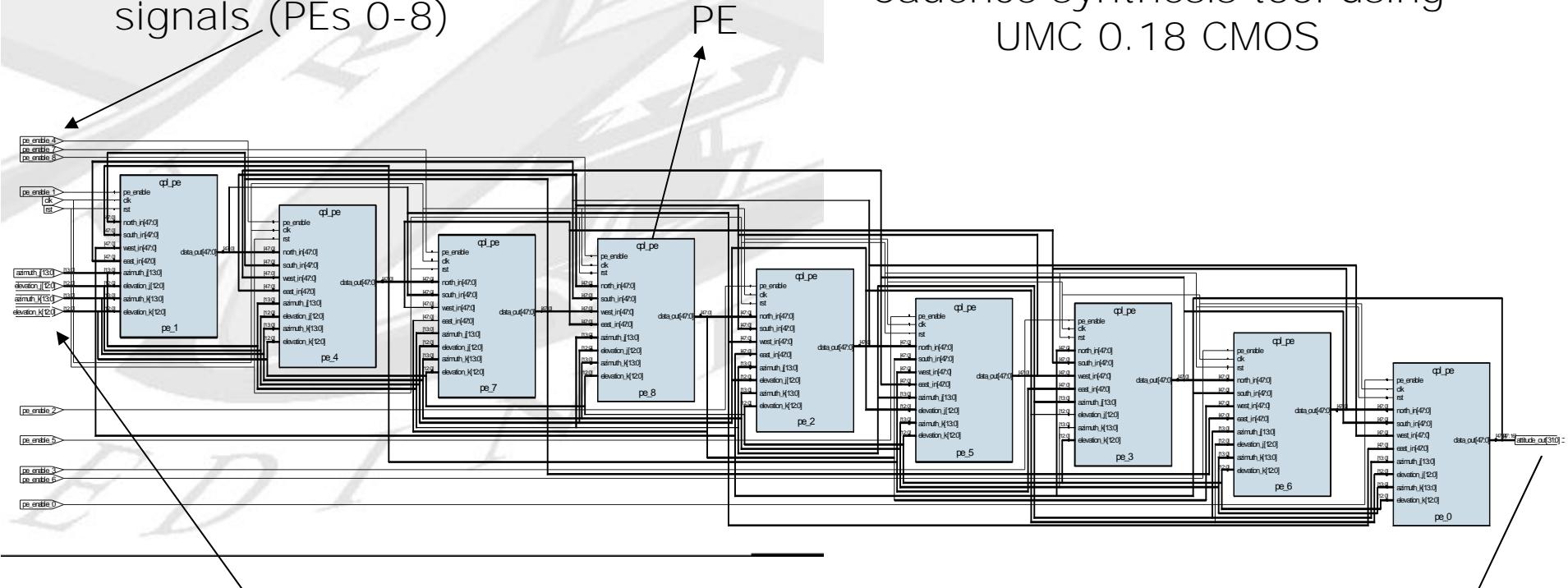
- § Initialize Individual  $X_1$
- § Evaluate  $X_1$
- § Exchange data with adjacent PEs
- § Selection of best chromosome  $f_M = \max(f^N, f^E, f^S, f^W)$
- § Crossover  $X_1 \rightarrow X_M$  (probability 100%)
- § Mutation
- § Evaluation of two children chromosomes  $X'$  &  $X''$
- § Ranking Chromosomes  $X_1 = \max(X_1, X', X'')$
- § GA checks for convergence

- § In PFGA architectures the *population* is partitioned into sub-populations that consist of one individual
- § Each *chromosome* encodes the attributes associated with the solution



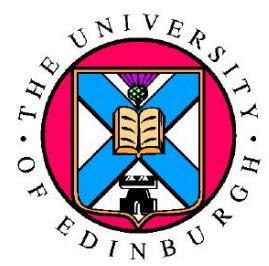
# CPL Array

Configuration  
signals (PEs 0-8)

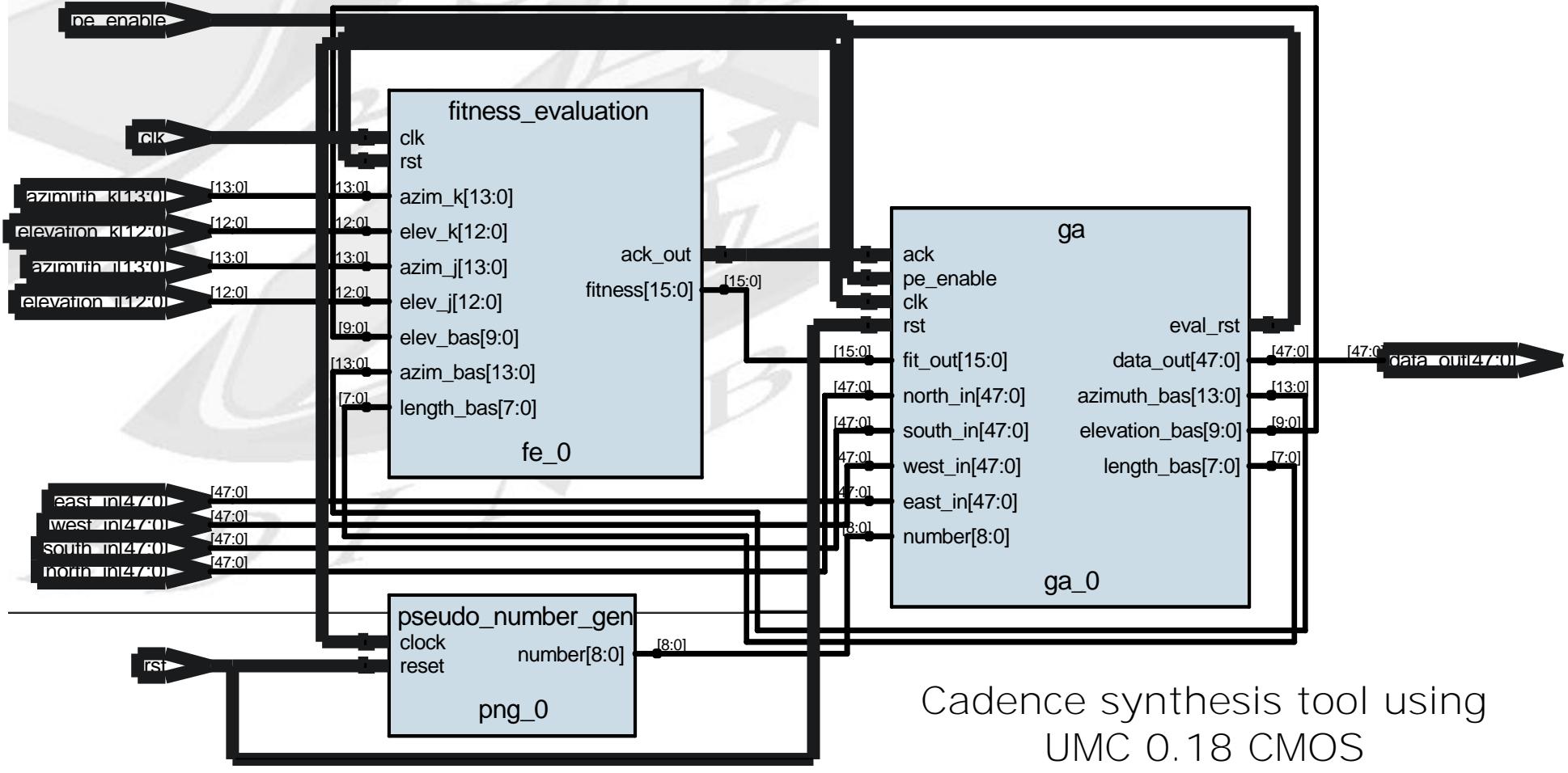


Azimuth & Elevation  
angles of two satellites

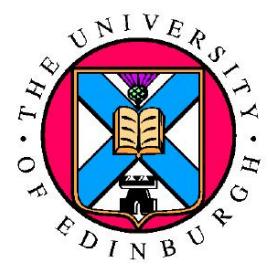
Cadence synthesis tool using  
UMC 0.18 CMOS



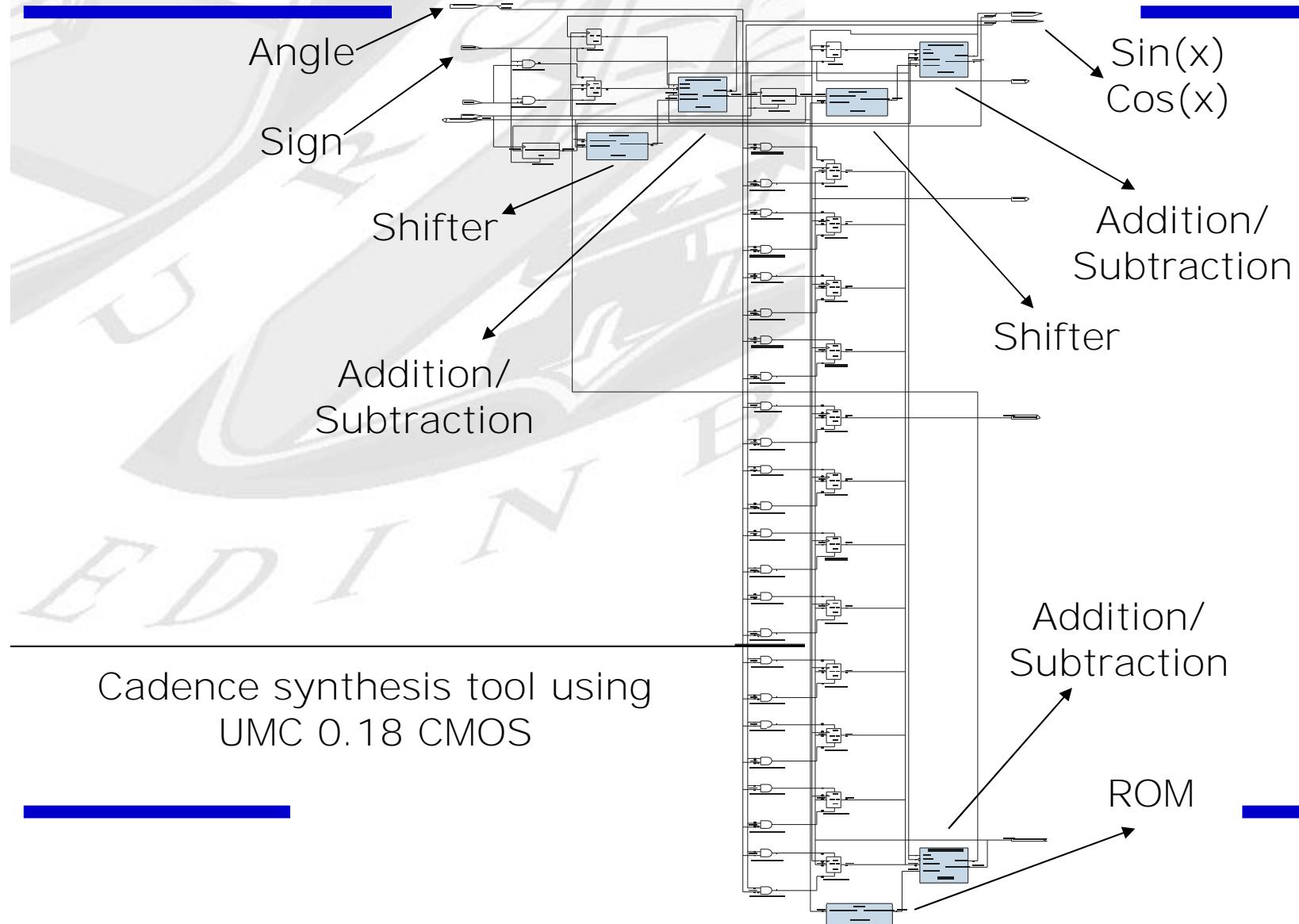
# PE Block Diagram



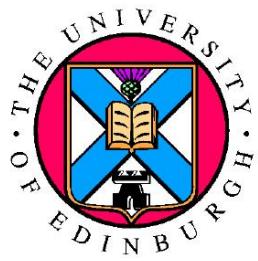
Cadence synthesis tool using  
UMC 0.18 CMOS



# Fitness Evaluation - CORDIC Unit

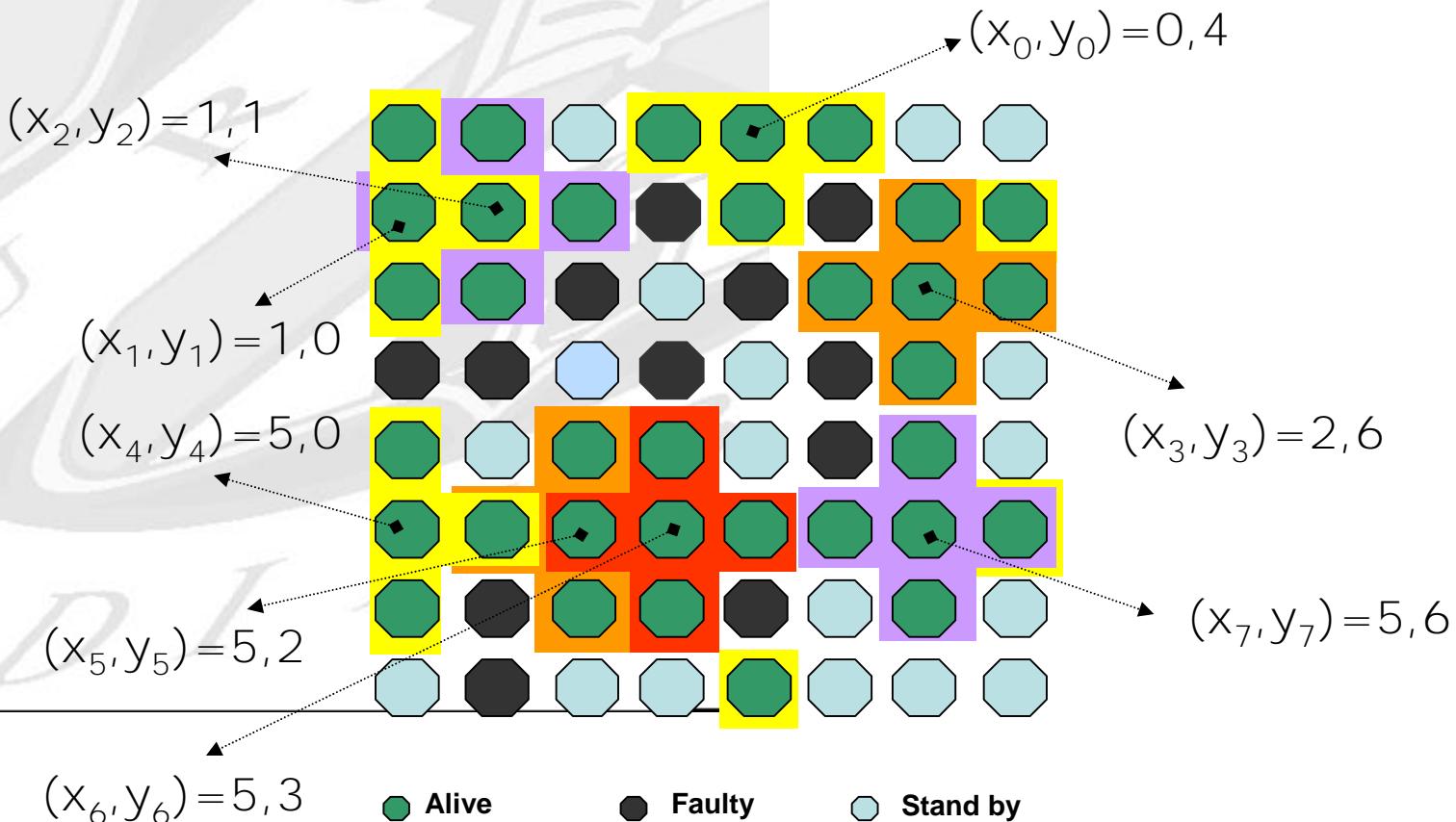


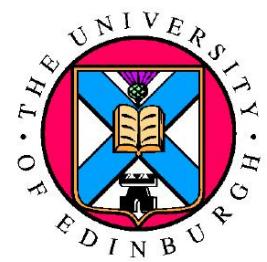
# Description of Control Layer (CTL)



- § It has the same architecture as the CPL and each controller monitors the performance (fitness-score) of the PEs in CPL
- § CTL deactivates for a certain time PEs in CPL that cannot converge to avoid potential degradation in the performance of the PGA

# Selection Approach



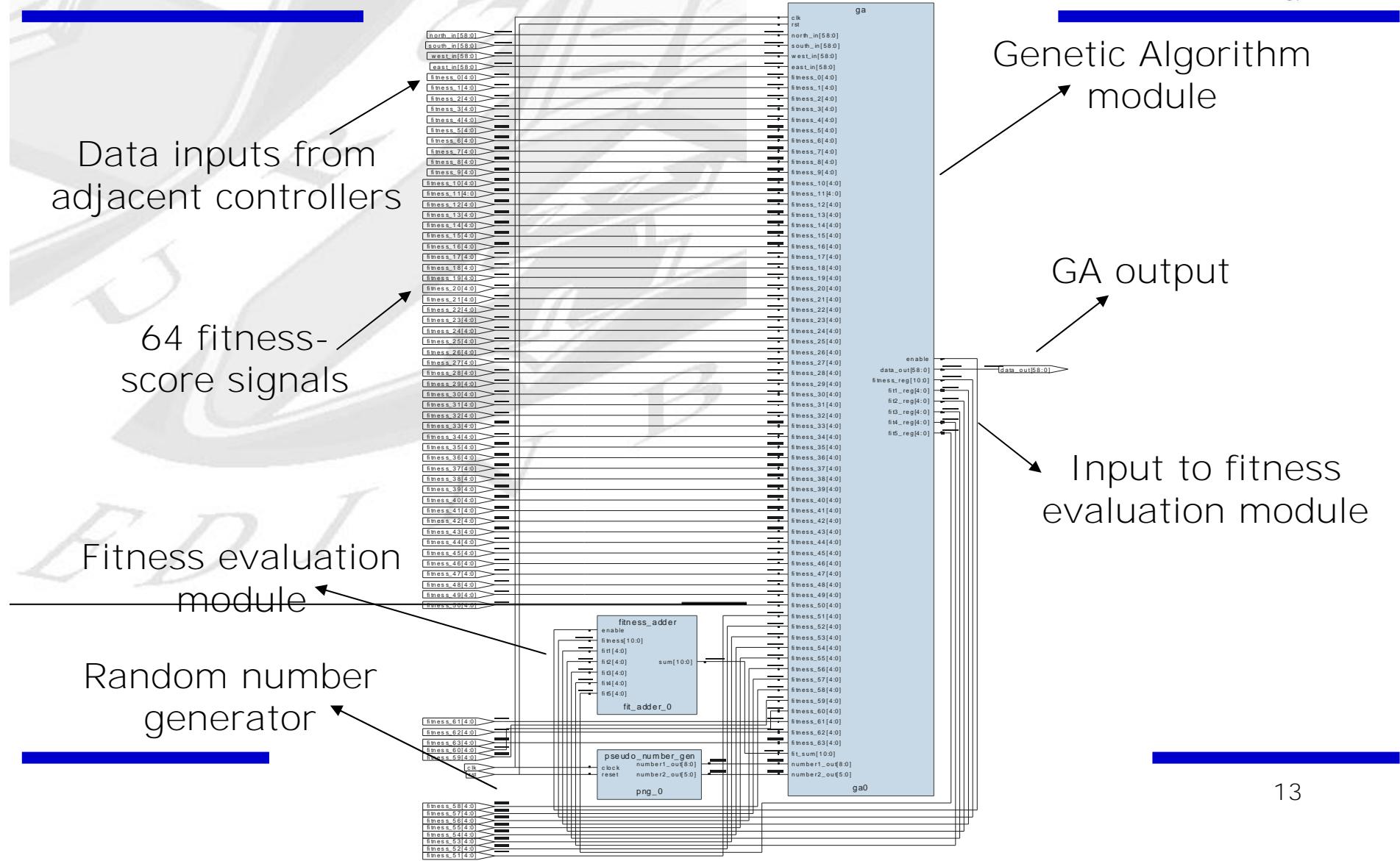


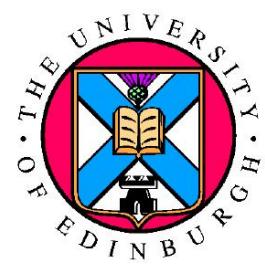
§ The coordinates of 8 central PEs of the selected crosses are encoded in the *chromosome* of each controller:

Coordinates PE_0	Coordinates PE_1	Coordinates PE_2	...	Coordinates PE_7
$(x_0, y_0)$ 6-bit	$(x_1, y_1)$ 6-bit	$(x_2, y_2)$ 6-bit		$(x_7, y_7)$ 6-bit

$$\text{Fitness-Score (CTL)} = \sum_{i=0}^7 \text{Fitness}_{\text{cross}_i} (\text{CPL})$$

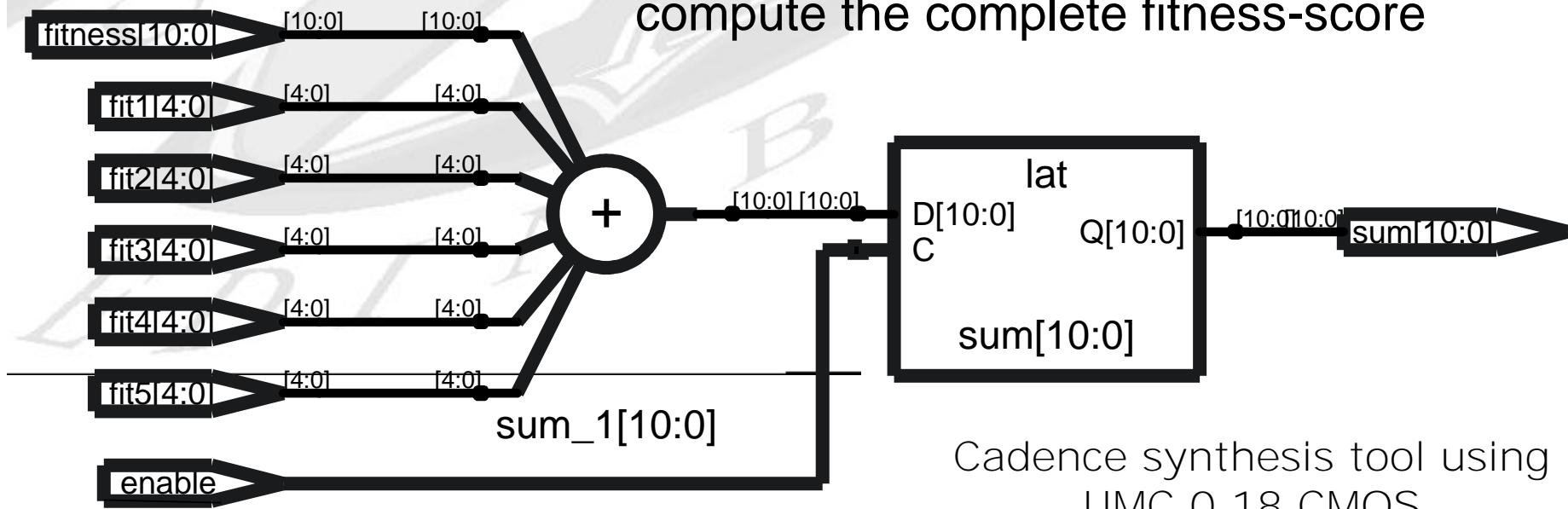
# Controller Design





# Fitness Calculation

- § At each clock-cycle the fitness-score of a cross is computed
- § The system needs 8 clock-cycles to compute the complete fitness-score

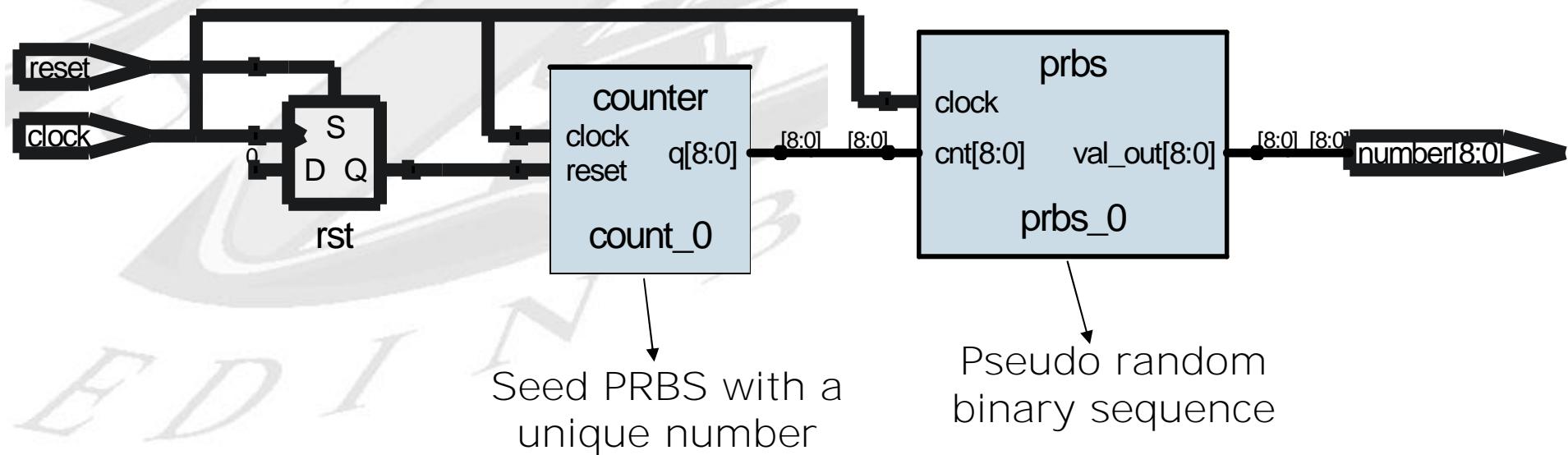


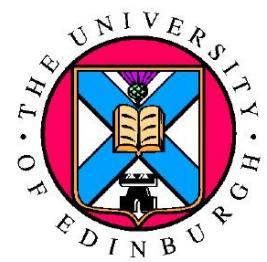
Cadence synthesis tool using  
UMC 0.18 CMOS

# Random Number Generator



Cadence synthesis tool using  
UMC 0.18 CMOS





# Faults Models

- § Single-Event-Upset (SEU) and Single-Event-Latchup (SEL) errors may occur in aerospace applications
- § In digital VLSI design this malfunction is translated to bit flips in memory elements
- § It is assumed that these appear as stuck-at faults occurring on the input/output registers or that these registers take erroneous values, due to the malfunction of other units

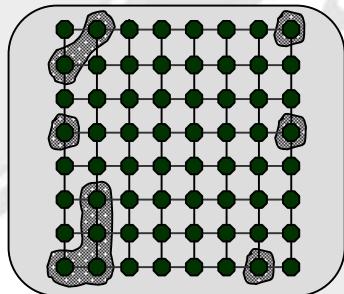


# Fault scenarios in the CPL

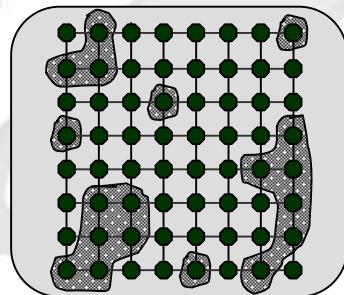
- § Two scenarios are considered for faults equal to 15, 30 and 40 percent of the total PEs
- § In the first scenario, CTL does not exist
- § In second, CTL is assumed to be immune to faults
- § Errors are modeled as stuck-at zero faults occurring on the input/output registers of certain PEs in the CPL



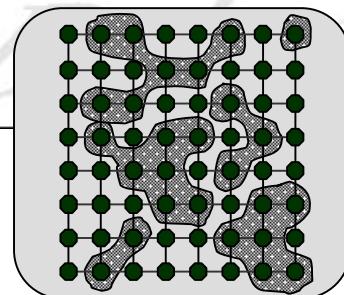
# Arrangement of faults



§ 15% of PEs are faulty



§ 30% of PEs are faulty

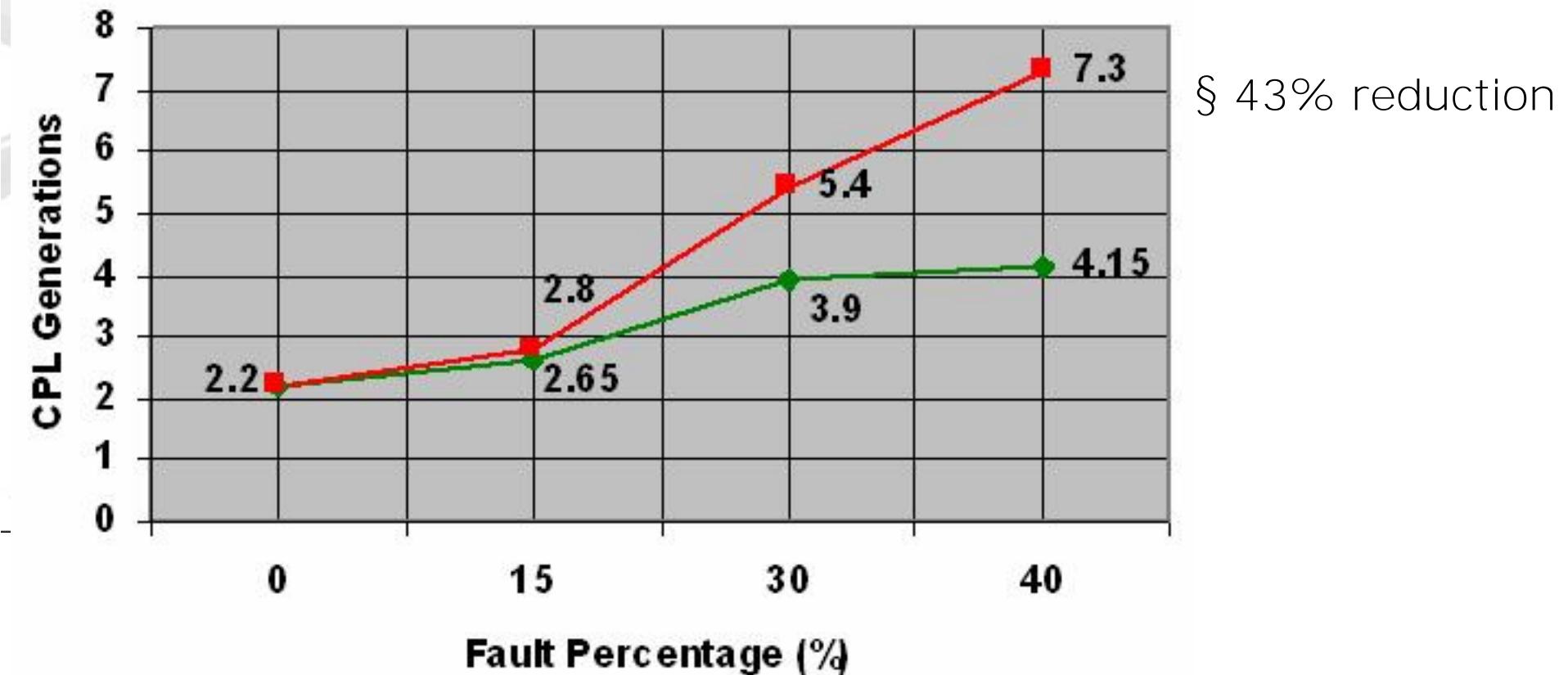


§ 40% of PEs are faulty



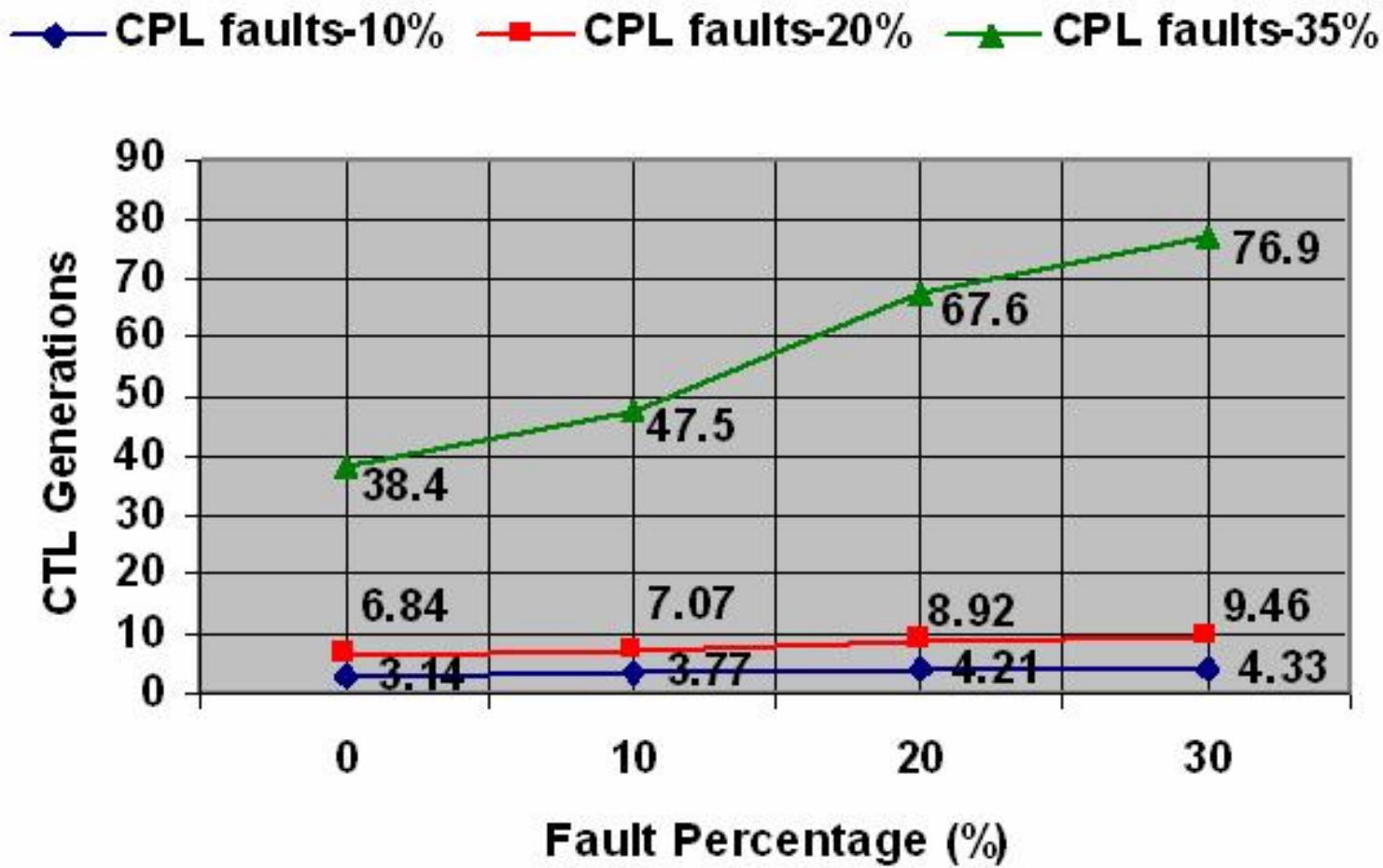
# Synthesized Netlist Simulation Results

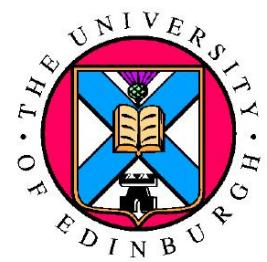
—●— with CTL      —■— without CTL





# Synthesized Netlist

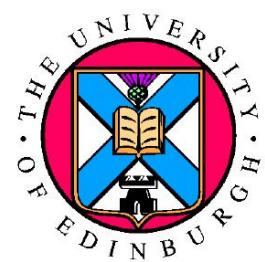




# Attitude update rate

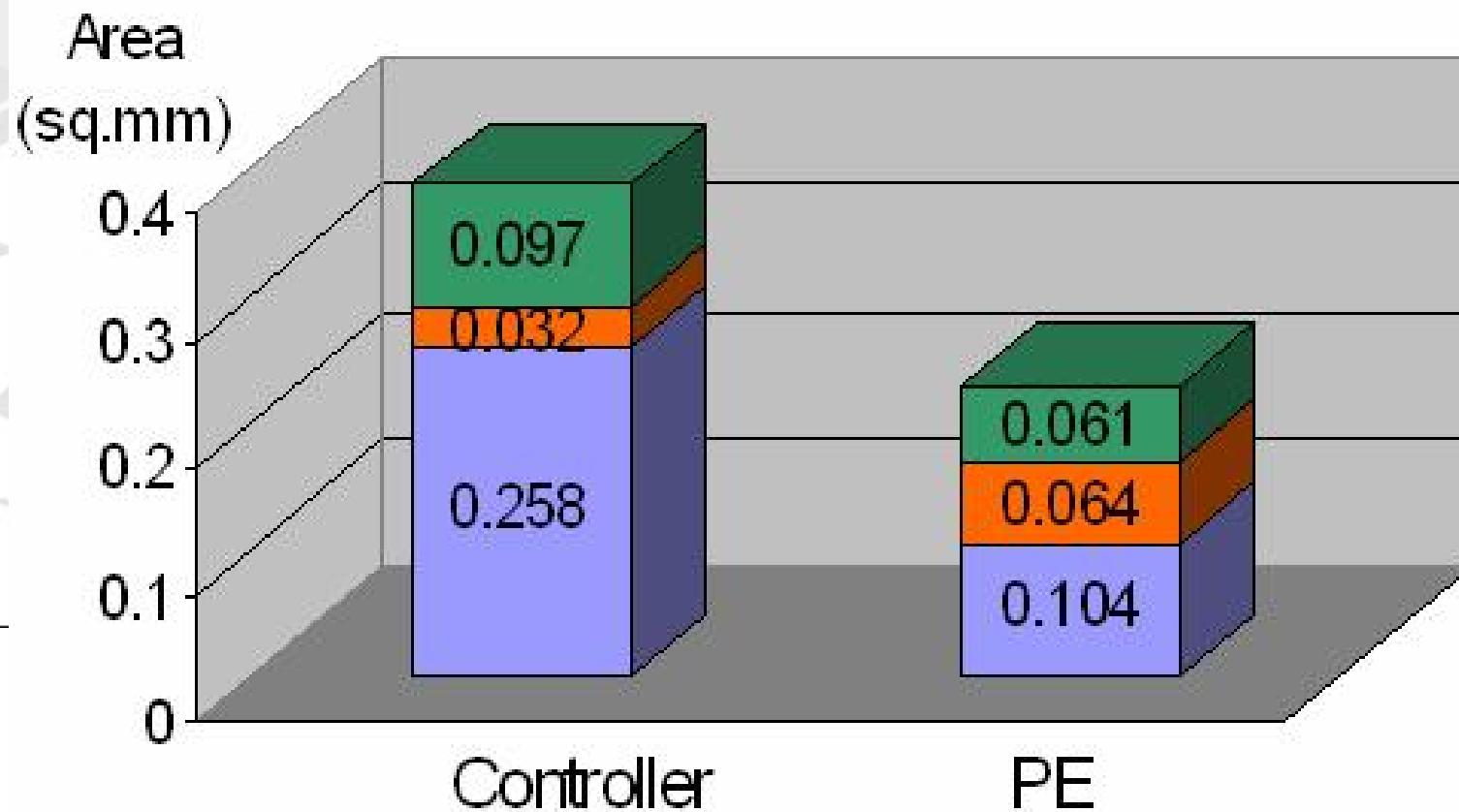
- § For the worst-case simulated scenario the system needs 76 and 3 generations for CTL and CPL respectively

	Clock-Cycles First Run	Clock-Cycles per Generation	Clock Period (nsec)	Time (usec)
CPL	358	241	50	54.05
CTL	57	40	50	154.85



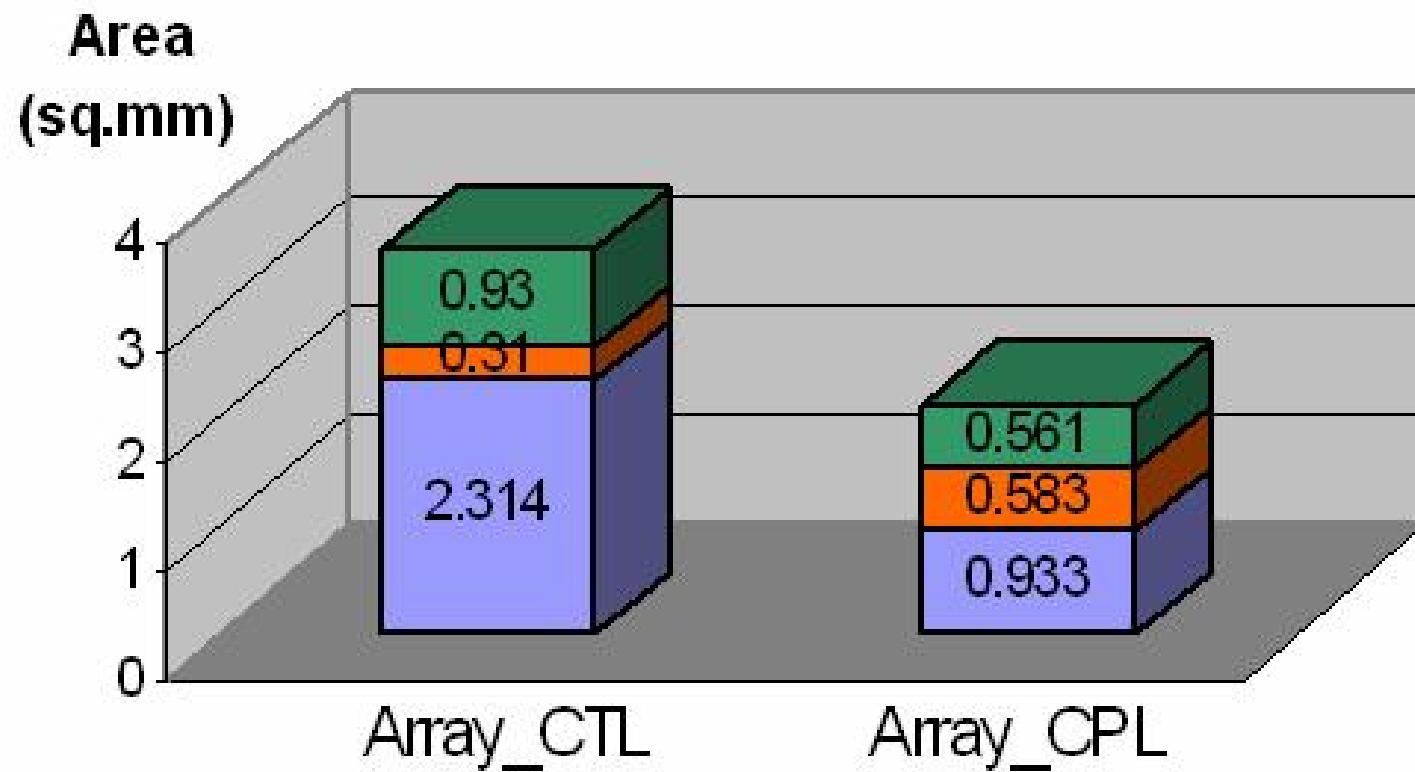
# Synthesis Results

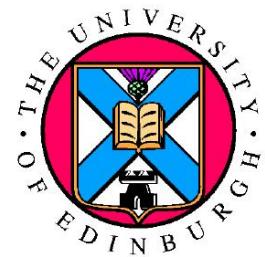
■ Combinational      ■ Non-combinational      ■ Net



# Synthesis Results (2)

■ Combinational      ■ Non-combinational      ■ Net





- § According to the obtained results the percentage of occurring faults can safely reach 35% & 30% for CPL and CTL respectively
- § Research has to be done into improving the fault models and investigating how faults affect the operation of the whole algorithm
- § More flexible routing schemes can be investigated in order to decrease the number of unusable PEs

